
# [ Анализ данных на языке SQL]

## День 2

## Вечер 4

www.specialist.ru

# Программа курса

- День 1. Работа с одной таблицей
- День 2. Работа с несколькими таблицами
- День 3. Другие вопросы (более сложные)

# День 2. Работа с несколькими таблицами (продолжение)

- Классификация подзапросов
- Сложные аспекты подзапросов
- Соединение таблиц (JOIN)
- Применение соединений
- Типичные ошибки при использовании соединений

# Классификация подзапросов

- Автономные и связанные (коррелированные)
- Скалярные и многозначные
- Могут применяться в разных местах запроса:
  - в качестве колонки,
  - в поле WHERE,
  - в списке IN,
  - в поле FROM вместо таблицы

# Сложные аспекты подзапросов

Когда таблицы напрямую не связаны друг с другом

(Допустимо несколько уровней вложенности)

```
-- Сколько штук товаров продано в каждой категории (название)?

SELECT CategoryName,
       (
       SELECT SUM(Quantity)
       FROM   [Order Details]
       WHERE  ProductID IN
              (
              SELECT ProductID
              FROM   Products
              WHERE  CategoryID = C.CategoryID
              )
       ) AS Штук
FROM   Categories C
```

Results | Messages

| | CategoryName | Штук |
|---|---|---|
| 1 | Beverages | 9532 |
| 2 | Condiments | 5298 |
| 3 | Confections | 7906 |
| 4 | Dairy Products | 9149 |
| 5 | Grains/Cereals | 4562 |
| 6 | Meat/Poultry | 4199 |
| 7 | Produce | 2990 |
| 8 | Seafood | 7681 |
| 9 | Сладкое | NULL |

# Сложные аспекты подзапросов (еще пример)

Сложность решения подзапросами – в структуре БД. Иногда основной список приходится создавать через трансфор - мацию (GROUP BY)

```
-- Сколько штук товаров продано в каждую страну?
SELECT      ShipCountry, Sum (SubTotal)
FROM        (
            SELECT  ShipCountry,
                    (
                    SELECT  Sum (Quantity)
                    FROM    [Order Details]
                    WHERE   OrderID = Orders.OrderID
                    ) AS SubTotal
            FROM    Orders
            ) MyOrders
GROUP BY    ShipCountry
```

100 %

Results | Messages

| | ShipCountry | (No column name) |
|---|---|---|
| 1 | Finland | 885 |
| 2 | USA | 9330 |
| 3 | Italy | 822 |
| 4 | Brazil | 4247 |
| 5 | Germany | 9213 |
| 6 | Switzerland | 1275 |

Query executed successfully. (local) (12.0 RTM) | CKO-ALM4NO8U7O1\Федор ... | Northwind | 00:00:00 | 21 rows

# Что такое JOIN?

- JOIN – это соединение таблиц
- Более сложный, более профессиональный, но более опасный
- Все начинается с простого CROSS JOIN
- Если UNION - это сложение таблиц , а JOIN - это скорее перемножение
- Согласовывать структуры не надо

# Какие виды соединений бывают

Классификация соединений:

- CROSS JOIN (перекрестное соединение)
- INNER JOIN (внутреннее соединение)
- OUTER JOIN (внешнее соединение)

  LEFT OUTER JOIN (левое внешнее соединение)
  RIGHT OUTER JOIN (правое внешнее соединение)
  FULL OUTER JOIN (полное внешнее соединение)

# Синтаксис запроса с соединением

- Синтаксис 1989 года

```
SELECT   T1.Col1, T2.Col2. T2.Col3
FROM    T1, T2
WHERE  T1.Col1 = T2.Col2
```

- Синтаксис 1992 года

```
SELECT   T1.Col1, T2.Col2. T2.Col3
FROM    T1 JOIN T2
     ON  T1.Col1 = T2.Col2
```

# Что такое перекрестное соединение (CROSS JOIN)

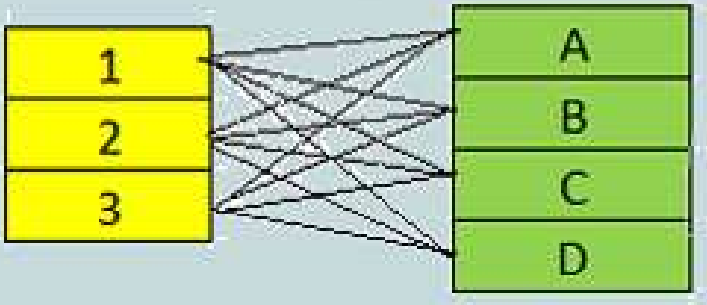
- Это простейший вид соединений и поэтому редко используемый в чистом виде
- Иногда его называют декартовым произведением 2х таблиц
- С него начинается любое соединение, а затем применяется фильтрация

```
SELECT *
FROM  T1 ,T2

-------------------------------

SELECT *
FROM  T1  CROSS JOIN  T2
```

# Декартово произведение (как работает JOIN)




$3 * 4 = 12$

| | |
|---|---|
| 1 | A |
| 1 | B |
| 1 | C |
| 1 | D |
| 2 | A |
| 2 | B |
| 2 | C |
| 2 | D |
| | |
| 3 | C |
| 3 | D |

# INNER JOIN (внутреннее соединение)

- INNER JOIN (внутреннее соединение)
- В результат попадают данные из 2-х таблиц, которые соответствуют условию соединения

```
SELECT *
FROM  T1 ,T2
WHERE T1.col1 = T2.col2
```

---

```
SELECT *
FROM  T1  INNER JOIN  T2
ON T1.col1 = T2.col2
```

# INNER JOIN (внутреннее соединение)

```
SELECT *
FROM  Клиенты  JOIN  Заказы
ON Клиенты.НомерКл = Заказы.НомерКл
```

Клиенты (левая таблица)

| | | |
|---|---|---|
| Вася | Пупкин | 1 |
| Петя | Иванов | 2 |
| Сережа | Николаев | 3 |

Заказы (правая таблица)

| | | |
|---|---|---|
| 1 | 1001 | 01.01.2019 |
| 1 | 1002 | 03.02.2019 |
| 3 | 1003 | 05.07.2019 |
| 4 | 1004 | 06.07.2019 |

| | | | | | |
|---|---|---|---|---|---|
| Вася | Пупкин | 1 | 1 | 1001 | 01.01.2019 |
| Вася | Пупкин | 1 | 1 | 1002 | 03.02.2019 |
| Сережа | Николае | 3 | 3 | 1003 | 05.07.2019 |

# LEFT OUTER JOIN (левое внешнее соединение)

- LEFT OUTER JOIN (левое внешнее соединение)
- Левое внешнее соединение – из левой таблицы в результат попадут все данные, а из правой – только те, которые соответствуют условию ON

```
SELECT  *
FROM   T1  LEFT OUTER JOIN  T2
ON       T1.col1 = T2.col2
```

# LEFT OUTER JOIN
# (левое внешнее соединение)

```
SELECT *
FROM  Клиенты LEFT JOIN Заказы
ON Клиенты.НомерКл = Заказы.НомерКл
```

Клиенты (левая таблица)

| | | |
|---|---|---|
| Вася | Пупкин | 1 |
| Петя | Иванов | 2 |
| Сережа | Николаев | 3 |

Заказы (правая таблица)

| | | |
|---|---|---|
| 1 | 1001 | 01.01.2019 |
| 1 | 1002 | 03.02.2019 |
| 3 | 1003 | 05.07.2019 |
| 4 | 1004 | 06.07.2019 |

| | | | | | |
|---|---|---|---|---|---|
| Вася | Пупкин | 1 | 1 | 1001 | 01.01.2019 |
| Вася | Пупкин | 1 | 1 | 1002 | 03.02.2019 |
| Сережа | Николаев | 3 | 3 | 1003 | 05.07.2019 |
| Петя | Иванов | 2 | NULL | NULL | NULL |

# RIGHT OUTER JOIN (правое внешнее соединение)

- RIGHT OUTER JOIN (правое внешнее соединение)
- Правое внешнее соединение – из правой таблицы в результат попадут все данные, а из левой – только те, которые соответствуют условию ON

```
SELECT  *
FROM   T1  RIGHT OUTER JOIN  T2
ON       T1.col1 = T2.col2
```

# RIGHT OUTER JOIN
# (правое внешнее соединение)

```
SELECT *
FROM  Клиенты RIGHT JOIN Заказы
ON Клиенты.НомерКл = Заказы.НомерКл
```

Клиенты (левая таблица)

| | | |
|---|---|---|
| Вася | Пупкин | 1 |
| Петя | Иванов | 2 |
| Сережа | Николаев | 3 |

Заказы (правая таблица)

| | | |
|---|---|---|
| 1 | 1001 | 01.01.2019 |
| 1 | 1002 | 03.02.2019 |
| 3 | 1003 | 05.07.2019 |
| 4 | 1004 | 06.07.2019 |

| | | | | | |
|---|---|---|---|---|---|
| Вася | Пупкин | 1 | 1 | 1001 | 01.01.2019 |
| Вася | Пупкин | 1 | 1 | 1002 | 03.02.2019 |
| Сережа | Николаев | 3 | 3 | 1003 | 05.07.2019 |
| NULL | NULL | NULL | 4 | 1004 | 06.07.2019 |

# FULL OUTER JOIN (полное внешнее соединение)

- FULL OUTER JOIN (полное внешнее соединение)
- Полное внешнее соединение – из обеих таблиц в результат попадут все данные (т.е. те – которые соответствуют условию ON, плюс те – которые не соответствуют слева, плюс те – которые не соответствуют справа

```
SELECT  *
FROM   T1  FULL OUTER JOIN  T2
ON       T1.col1 = T2.col2
```

# FULL OUTER JOIN
# (полное внешнее соединение)

```
SELECT *
FROM  Клиенты FULL JOIN Заказы
ON Клиенты.НомерКл = Заказы.НомерКл
```

Клиенты (левая таблица)

| | | |
|---|---|---|
| Вася | Пупкин | 1 |
| Петя | Иванов | 2 |
| Сережа | Николаев | 3 |

Заказы (правая таблица)

| | | |
|---|---|---|
| 1 | 1001 | 01.01.2019 |
| 1 | 1002 | 03.02.2019 |
| 3 | 1003 | 05.07.2019 |
| 4 | 1004 | 06.07.2019 |

| | | | | | |
|---|---|---|---|---|---|
| Вася | Пупкин | 1 | 1 | 1001 | 01.01.2019 |
| Вася | Пупкин | 1 | 1 | 1002 | 03.02.2019 |
| Сережа | Николае | 3 | 3 | 1003 | 05.07.2019 |
| Петя | Иванов | 2 | NULL | NULL | NULL |
| NULL | NULL | NULL | 4 | 1004 | 06.07.2019 |

# Итого (внешние соединения)

- Левое внешнее соединение – из левой таблицы в результат попадут все данные, а из правой – только те, которые соответствуют условию ON
- Правое внешнее соединение – из правой таблицы попадут все данные, а из левой – только те, которые соответствуют условию ON
- Полное внешнее соединение – из обеих таблиц в результат попадут все данные (т.е. те – которые соответствуют условию ON, плюс те – которые не соответствуют слева, плюс те – которые не соответствуют справа

# Типичные ошибки при использовании соединений

- JOINs – имеют много преимуществ
  - компактность
  - тесную связь с реляционной моделью
  - производительность
- ! Требуют очень внимательного применения, так как могут порождать не совсем правильные результаты (или совсем не правильные)
- Типичные ошибки рассмотрим на примерах
- Начнем с простейшей задачи:

«Сколько заказов оформил каждый продавец в Лондон?»

# Проблемы при использовании соединений

Задача:

- Сколько заказов оформил каждый продавец в Лондон?

Теперь усложним:

- Сколько продуктов оформил каждый продавец в Лондон?

Теперь еще:

- Сколько штук продукта 1 оформил каждый продавец в Лондон?

И наконец:

- Сколько штук продукта, с названием ‘Chai’ оформил каждый продавец в Лондон?

# Типичные ошибки при использовании соединений

3 ошибки

- 1. Показать сколько заказов сделал каждый покупатель? (без LEFT JOIN – можно потерять покупателей, не делавших заказы)
- 2. LEFT JOIN может породить типовую ошибку 2 –> Diego Roel и Marie Bertrand почему то сделали по 1 заказу (NULL – засчитан как заказ)
- Ошибка 3 - страшная ошибка – очень коварная, ее легко исправить, но очень трудно найти (неправильный подсчет агрегатных значений)

Задача: Сколько заказов оформил каждый продавец в Лондон?

# Вопросы?